Title	Folded Preconditioner: A New Class of Preconditioners for Krylov Subspace Methods to Solve Redundancy-Reduced Linear Systems of Equations
Author(s)	Mifune, Takeshi; Takahashi, Yasuhito; Iwashita, Takeshi
Citation	IEEE TRANSACTIONS ON MAGNETICS (2009), 45(5): 2068-2075
Issue Date	2009-05
URL	http://hdl.handle.net/2433/109809
Right	© 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Type	Journal Article
Textversion	publisher

Folded Preconditioner: A New Class of Preconditioners for Krylov Subspace Methods to Solve Redundancy-Reduced Linear Systems of Equations

Takeshi Mifune[1], Yasuhito Takahashi[2], and Takeshi Iwashita[3]

[1]Department of Electrical Engineering, Kyoto University, Kyoto 615-8510, Japan
[2]Department of System Science, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
[3]Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606-8501, Japan

The A-φ formulation, which is widely used in electromagnetic analysis, leads to a redundant linear system of equations that includes a substantial number of redundant degrees of freedom (DOF). We can derive a redundancy-reduced linear system of equations by eliminating the redundant DOF, thereby decreasing the computation costs per iteration for iterative solvers, such as the incomplete Cholesky conjugate gradient (ICCG) solver. This does not, however, result in a reduction in total computation time, due to significant convergence deterioration. In this paper, we present a solution to this problem in the form of folded preconditioners. First, the theorem presented reveals that, for any preconditioned Krylov subspace method for the original redundant linear systems, we can derive the equivalent Krylov subspace method for the redundancy-reduced linear systems by using the corresponding folded preconditioner. As an uncomplicated example, the standard ICCG solver for the original redundant systems has exactly the same convergence property as the CG solver for the redundancy-reduced systems using the folded variant of the IC preconditioner (the folded IC preconditioner). Furthermore, we discuss efficient computational procedures for the folded preconditioners and the design of Krylov subspace algorithms using the preconditioners. A sample full-wave analysis demonstrates the good performance of a newly developed solver, the conjugate orthogonal conjugate gradient (COCG) method with the folded IC preconditioner. The new solver not only lowers the computation costs per iteration by reducing the number of DOF, but also completely avoids the convergence deterioration.

Index Terms—**A-φ method, Krylov subspace method, preconditioning, singular linear system of equations.**

I. Introduction

In electromagnetic finite element (FE) analysis, the FE formulation leads to a linear system of equations that includes a substantial number of degrees of freedom (DOF). Whereas fast linear solvers are needed to speed-up the analysis, the number of DOF becomes extremely large in a large-scale analysis, thus prohibiting the use of direct solvers. Possible alternatives are the iterative methods [1], [2], such as the preconditioned Krylov subspace (KS) methods.

Certain numerical formulations used in electromagnetic field analysis lead to redundant (singular) linear systems of equations, which involve redundant DOF. A popular example is the A-φ (E-φ) formulation [3], [4], which is widely used in magneto-quasi-static or full-wave analysis. The A-φ formulation leads to a linear system of equations, in which the DOF associated with the scalar potential φ are redundant. The redundant DOF can be eliminated, for instance, by simply setting them to zero. Another example is the A-formulation [4], [5] used in magnetostatic analysis. The dimensions of the linear system of equations arising from the A-formulation can be reduced by using a tree-cotree gauge [6].

It is easy to reduce the number of redundant DOF in these formulations, and this results in a reduction in computation costs per iteration for the iterative solvers. However, in many practical applications, it is preferable to solve the redundant linear systems of equations without a reduction in DOF, because the redundancy reduction frequently causes significant deterioration in the convergence property of the iterative solvers [7], [8]. References [8]–[11] provide an explanation for the poor convergence caused by the redundancy reduction.

In this paper, we focus on devising an efficient preconditioned KS method for a redundancy-reduced linear system of equations that attains similar convergence to the original preconditioned KS method for the redundant linear system of equations. For this purpose, we propose a new class of preconditioners for the KS methods applied to the redundancy-reduced linear systems of equations: *the folded preconditioners*. The theorem we present implies that, for any preconditioned KS method for a redundant linear system of equations, we can construct a mathematically equivalent KS method for the redundancy-reduced linear system of equations by using the new preconditioner. For example, we can confirm that the incomplete Cholesky conjugate gradient (ICCG) solver for a redundant linear system of equations is equivalent to the conjugate gradient (CG) solver for the redundancy-reduced linear system of equations using the *folded* variant of the IC preconditioner. Both solvers have the same convergence property mathematically.

For any preconditioner for the original redundant linear system of equations, the corresponding folded preconditioner can be derived. Using the folded preconditioner, we can reduce the redundant DOF and the computation costs with respect to the matrix-vector multiplications and other vector computations in the KS algorithm without any deterioration in the convergence property. Moreover, we discuss the construction of new KS solvers using the folded preconditioners together with efficient implementations of the special preconditioners. Numerical tests confirm that the proposed methods are efficient.

II. Redundancy-Reduced Krylov Subspace Methods and Folded Preconditioners

A. Notation

Given that K is a positive definite matrix, the notation $(\boldsymbol{x}, \boldsymbol{y})_K = \boldsymbol{x}^H K \boldsymbol{y}$ denotes the inner product. Omission of the

Manuscript received September 09, 2008; revised January 10, 2009. Current version published April 17, 2009. Corresponding author: T. Mifune (e-mail: mifune@fem.kuee.kyoto-u.ac.jp).

Digital Object Identifier 10.1109/TMAG.2009.2014156

subscript means that K is a identity matrix, i.e., $(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^H\boldsymbol{y}$. Similarly, $||\boldsymbol{x}||_K$ and $||\boldsymbol{x}||$ denote vector norms, i.e., $(\boldsymbol{x},\boldsymbol{x})_K^{1/2}$ and $(\boldsymbol{x},\boldsymbol{x})^{1/2}$, respectively.

The identity matrices are denoted by I.

B. Linear System of Equations Including Redundancy

The linear system of equations considered here is given by

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A \in \boldsymbol{C}^{N\times N}, \quad \boldsymbol{x} \in \boldsymbol{C}^N, \quad \boldsymbol{b} \in \boldsymbol{C}^N \tag{1}$$

where $A, \boldsymbol{x}, \boldsymbol{b}$, and N are the coefficient matrix, unknown vector, right-hand side (RHS) vector, and a positive integer, respectively. Matrix A is meant to be a sparse matrix, but the theorem we present still holds even if A is not sparse.

Suppose that (1) is a singular system of equations, i.e., $\mathrm{rank}(A) < N$. The coefficient matrix in (1) can be written as shown below, after an appropriate reordering, such that $\mathrm{rank}(A) = \mathrm{rank}(A_\mathrm{r})$

$$A = \begin{pmatrix} A_\mathrm{r} & A_\mathrm{r}B \\ CA_\mathrm{r} & CA_\mathrm{r}B \end{pmatrix}, \quad A_\mathrm{r} \in \boldsymbol{C}^{L\times L}, \quad B \in \boldsymbol{C}^{L\times(N-L)}, \quad C \in \boldsymbol{C}^{(N-L)\times L} \tag{2}$$

where L is an integer that satisfies $\mathrm{rank}(A) \le L < N$.

Consequently, the unknown and RHS vectors are written as

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \quad \boldsymbol{x}_1 \in \boldsymbol{C}^L, \quad \boldsymbol{x}_2 \in \boldsymbol{C}^{(N-L)} \tag{3}$$

$$\boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_\mathrm{r} \\ C\boldsymbol{b}_\mathrm{r} \end{pmatrix}, \quad \boldsymbol{b}_\mathrm{r} \in \boldsymbol{C}^L. \tag{4}$$

The rightmost equality in (4) should be satisfied to ensure the existence of a solution.

Given that

$$\boldsymbol{x}_\mathrm{r} = \boldsymbol{x}_1 + B\boldsymbol{x}_2 \tag{5}$$

the following reduced linear system of equations is derived from (1)

$$A_\mathrm{r}\boldsymbol{x}_\mathrm{r} = \boldsymbol{b}_\mathrm{r}. \tag{6}$$

Once a solution for (6) has been obtained, one of the solutions in (1) can be obtained from

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_\mathrm{r} \\ 0 \end{pmatrix}. \tag{7}$$

From the viewpoint of the number of DOF and the number of nonzero entries in the coefficient matrix, it appears to be more attractive to solve (6) than to solve (1). However, when conventional iterative solvers are applied, the redundancy reduction causes a significant deterioration in the convergence property [7], [8]. Consequently, the elimination of redundant DOF does not contribute to a reduction in the total computation time in practical applications, such as eddy-current or full-wave analysis using the A-φ formulation.

C. Preconditioned Krylov Subspace Methods

By applying right and left preconditionings to (1), we obtain the preconditioned linear system of equations

$$M_1AM_2\boldsymbol{y} = M_1\boldsymbol{b} \tag{8}$$

where

$$\boldsymbol{y} = M_2^{-1}\boldsymbol{x}. \tag{9}$$

Consider applying a specific KS method to (8). The KS method generates the following approximate solution for $\boldsymbol{y}$ at the nth iteration

$$\boldsymbol{y}^{(n)} = \boldsymbol{y}^{(0)} + \boldsymbol{v}^{(n)} \tag{10}$$

$$\boldsymbol{v}^{(n)} \in \mathrm{span}\left\{M_1\boldsymbol{r}^{(0)}, M_1AM_2M_1\boldsymbol{r}^{(0)}, \ldots, M_1(AM_2M_1)^{n-1}\boldsymbol{r}^{(0)}\right\} \tag{11}$$

where $\boldsymbol{y}^{(0)}$ is an initial approximation for $\boldsymbol{y}$ and $\boldsymbol{r}^{(i)}$ denotes the residual vector with respect to (1), i.e.,

$$\boldsymbol{r}^{(i)} = \boldsymbol{b} - AM_2\boldsymbol{y}^{(i)}. \tag{12}$$

Consequently, the corresponding approximate solutions $\boldsymbol{x}^{(n)} = M_2\boldsymbol{y}^{(n)}$ are given by

$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^{(0)} + \boldsymbol{u}^{(n)} \tag{13}$$

$$\boldsymbol{u}^{(n)} \in \mathrm{span}\left\{M\boldsymbol{r}^{(0)}, MAM\boldsymbol{r}^{(0)}, \ldots, M(AM)^{n-1}\boldsymbol{r}^{(0)}\right\} \tag{14}$$

where $M = M_2M_1$.

In the following discussion, suppose that M (the inverse of the preconditioner) is expressed as

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, \quad M_{11} \in \boldsymbol{C}^{L\times L},\ M_{12} \in \boldsymbol{C}^{L\times(N-L)}, \quad M_{21} \in \boldsymbol{C}^{(N-L)\times L},\ M_{22} \in \boldsymbol{C}^{(N-L)\times(N-L)}. \tag{15}$$

Similarly to (13) and (14), we can illustrate that a preconditioned KS method for (6) generates the approximate solutions

$$\boldsymbol{x}_\mathrm{r}^{(n)} = \boldsymbol{x}_\mathrm{r}^{(0)} + \boldsymbol{u}_\mathrm{r}^{(n)} \tag{16}$$

$$\boldsymbol{u}_\mathrm{r}^{(n)} \in \mathrm{span}\left\{M_\mathrm{r}\boldsymbol{r}_\mathrm{r}^{(0)}, M_\mathrm{r}A_\mathrm{r}M_\mathrm{r}\boldsymbol{r}_\mathrm{r}^{(0)}, \ldots, M_\mathrm{r}(A_\mathrm{r}M_\mathrm{r})^{n-1}\boldsymbol{r}_\mathrm{r}^{(0)}\right\} \tag{17}$$

where the L by L matrix M_r and L-dimensional vector $\boldsymbol{r}_\mathrm{r}^{(i)}$ denote, respectively, the inverse of the preconditioner and the following residual vector with respect to (6)

$$\boldsymbol{r}_\mathrm{r}^{(i)} = \boldsymbol{b}_\mathrm{r} - A_\mathrm{r}\boldsymbol{x}_\mathrm{r}^{(i)}. \tag{18}$$

In this paper, to distinguish between the methods, we call the preconditioned KS methods for (6), redundancy-reduced KS (RR-KS) methods.

D. Theorem Relating to Redundancy Reduction

When a preconditioned KS method expressed by (13) and (14) is given, the following sequence of vectors can be regarded as approximate solutions for $\boldsymbol{x}_{\mathrm{r}}$

$$\boldsymbol{x}_{\mathrm{r}}'^{(n)} = \boldsymbol{x}_1^{(n)} + B\boldsymbol{x}_2^{(n)} \tag{19}$$

where

$$\boldsymbol{x}^{(n)} = \begin{pmatrix} \boldsymbol{x}_1^{(n)} \\ \boldsymbol{x}_2^{(n)} \end{pmatrix}, \quad \boldsymbol{x}_1^{(n)} \in \boldsymbol{C}^L, \quad \boldsymbol{x}_2^{(n)} \in \boldsymbol{C}^{N-L}. \tag{20}$$

Furthermore, the residual vector with respect to (1) has the following form:

$$\boldsymbol{r}^{(n)} = \boldsymbol{b} - A\boldsymbol{x}^{(n)} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} \tag{21}$$

where

$$\boldsymbol{r}_{\mathrm{r}}'^{(n)} = \boldsymbol{b}_{\mathrm{r}} - A_{\mathrm{r}}\boldsymbol{x}_{\mathrm{r}}'^{(n)}. \tag{22}$$

In this paper, when the preconditioned KS methods for (1) and (6) generate identical approximate solution vectors, i.e., for all n

$$\boldsymbol{x}_{\mathrm{r}}^{(n)} = \boldsymbol{x}_{\mathrm{r}}'^{(n)} \tag{23}$$

the two methods are termed *equivalent*. *Equivalent* solvers have the same convergence property, in the sense that

$$\boldsymbol{b} - A\begin{pmatrix} \boldsymbol{x}_{\mathrm{r}}^{(n)} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}^{(n)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} = \boldsymbol{r}^{(n)}. \tag{24}$$

(See (7) and (21).)

Now, we derive a crucial theorem.

Theorem 1: For any preconditioned KS method for (1), there is an *equivalent* preconditioned RR-KS method, in which the following matrix M_{f} is used as M_{r} in (17)

$$M_{\mathrm{f}} = M_{11} + M_{12}C + BM_{21} + BM_{22}C. \tag{25}$$

Proof: Since

$$\begin{aligned} AM\boldsymbol{r}^{(0)} &= \begin{pmatrix} A_{\mathrm{r}} & A_{\mathrm{r}}B \\ CA_{\mathrm{r}} & CA_{\mathrm{r}}B \end{pmatrix} \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(0)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(0)} \end{pmatrix} \\ &= \begin{pmatrix} A_{\mathrm{r}}M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(0)} \\ CA_{\mathrm{r}}M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(0)} \end{pmatrix} \end{aligned} \tag{26}$$

we can recursively obtain

$$(AM)^i\boldsymbol{r}^{(0)} = \begin{pmatrix} (A_{\mathrm{r}}M_{\mathrm{f}})^i\boldsymbol{r}_{\mathrm{r}}'^{(0)} \\ C(A_{\mathrm{r}}M_{\mathrm{f}})^i\boldsymbol{r}_{\mathrm{r}}'^{(0)} \end{pmatrix} \tag{27}$$

$$M(AM)^i\boldsymbol{r}^{(0)} = \begin{pmatrix} (M_{11} + M_{12}C)(A_{\mathrm{r}}M_{\mathrm{f}})^i\boldsymbol{r}_{\mathrm{r}}'^{(0)} \\ (M_{21} + M_{22}C)(A_{\mathrm{r}}M_{\mathrm{f}})^i\boldsymbol{r}_{\mathrm{r}}'^{(0)} \end{pmatrix}. \tag{28}$$

By substituting (28) into (14) and noting (19) and (20), we can derive

$$\boldsymbol{x}_{\mathrm{r}}'^{(n)} = \boldsymbol{x}_{\mathrm{r}}'^{(0)} + \boldsymbol{u}_{\mathrm{r}}'^{(n)} \tag{29}$$

$$\boldsymbol{u}_{\mathrm{r}}'^{(n)} \in \mathrm{span}\left\{ M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(0)}, M_{\mathrm{f}}A_{\mathrm{r}}M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(0)}, \ldots, M_{\mathrm{f}}(A_{\mathrm{r}}M_{\mathrm{f}})^{n-1}\boldsymbol{r}_{\mathrm{r}}'^{(0)} \right\}. \tag{30}$$

Finally, consider an RR-KS method expressed by (16) and (17). By choosing $\boldsymbol{x}_{\mathrm{r}}^{(0)} = \boldsymbol{x}_{\mathrm{r}}'^{(0)}$, we have

$$\boldsymbol{r}_{\mathrm{r}}^{(0)} = \boldsymbol{r}_{\mathrm{r}}'^{(0)}. \tag{31}$$

Because the subspace in (17) is identical to the one in (30) when $M_{\mathrm{r}} = M_{\mathrm{f}}$, we are able to construct an *equivalent* RR-KS method, by choosing $\boldsymbol{u}_{\mathrm{r}}^{(n)}$ that satisfies

$$\boldsymbol{u}_{\mathrm{r}}^{(n)} = \boldsymbol{u}_{\mathrm{r}}'^{(n)}. \tag{32}$$

□

We call the special preconditioners using (25) *the folded preconditioners*.

III. Construction of RR-KS Algorithms

A. Strategies Used to Construct RR-KS Algorithms With the Folded Preconditioners

According to Theorem 1, for an arbitrary preconditioned KS method applicable to a redundant system of equations, we can find at least one RR-KS method using the corresponding folded preconditioner that has the same convergence property as the original KS method. In other words, the folded preconditioner enables us to avoid any deterioration in convergence caused by the redundancy reduction.

The tasks required to develop new solvers using the folded preconditioners include choosing concrete RR-KS algorithms that generate approximate solutions from the affine space represented by (16) and (17), and designing efficient computational procedures for the special preconditioner.

Here, we present two strategies for constructing concrete RR-KS algorithms with the folded preconditioners. These two strategies do not always produce the same result.

1) Strategy to Derive the Equivalent Algorithm: Equation (23) [and (32)] immediately presents a strategy, that derives an equivalent algorithm from a specific preconditioned KS solver for (1), such as the CG algorithm [12].

By choosing approximate solutions such that (23) holds, we can generate the equivalent RR-KS algorithm that has consistently the same convergence property as the original solver. This property may be attractive when the original KS solver is well-established.

As mentioned below, deriving the equivalent RR-KS algorithm is accomplished by cautiously replacing some of vectors based on (19). A drawback of this strategy is that the generated algorithm does not always take a simple and sophisticated form, which is not the case with the second strategy.

2) Simple Strategy Using the Folded Preconditioner: If we do not adhere to the strict equivalence mentioned above, a different strategy may be more efficient.

Once we have decided to use a specific folded preconditioner, a simple and natural way to search for a good approximate solution from the affine space described in (16) and (17) is to utilize a known (sophisticated) KS method, such as the CG method.

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}, \quad \beta_{-1} = 0$$
for $n = 0, 1, \ldots$
$$\boldsymbol{p}^{(n)} = M\boldsymbol{r}^{(n)} + \beta^{(n-1)}\boldsymbol{p}^{(n-1)}$$
$$\alpha^{(n)} = \left(M\boldsymbol{r}^{(n)}, \boldsymbol{r}^{(n)}\right) \Big/ \left(\boldsymbol{p}^{(n)}, A\boldsymbol{p}^{(n)}\right)$$
$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha^{(n)}\boldsymbol{p}^{(n)}$$
$$\boldsymbol{r}^{(n+1)} = \boldsymbol{r}^{(n)} - \alpha^{(n)}A\boldsymbol{p}^{(n)}$$
$$\beta^{(n)} = \left(M\boldsymbol{r}^{(n+1)}, \boldsymbol{r}^{(n+1)}\right) \Big/ \left(M\boldsymbol{r}^{(n)}, \boldsymbol{r}^{(n)}\right)$$
endfor

Fig. 1. Preconditioned CG algorithm for (1).

$$\boldsymbol{r}_{\mathrm{r}}^{(0)} = \boldsymbol{b}_{\mathrm{r}} - A_{\mathrm{r}}\boldsymbol{x}_{\mathrm{r}}^{(0)}, \quad \beta_{-1} = 0$$
for $n = 0, 1, \ldots$
$$\boldsymbol{p}_{\mathrm{r}}^{(n)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n)} + \beta^{(n-1)}\boldsymbol{p}_{\mathrm{r}}^{(n-1)}$$
$$\alpha^{(n)} = \left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n)}, \boldsymbol{r}_{\mathrm{r}}^{(n)}\right) \Big/ \left(\boldsymbol{p}_{\mathrm{r}}^{(n)}, A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}\right)$$
$$\boldsymbol{x}_{\mathrm{r}}^{(n+1)} = \boldsymbol{x}_{\mathrm{r}}^{(n)} + \alpha^{(n)}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\boldsymbol{r}_{\mathrm{r}}^{(n+1)} = \boldsymbol{r}_{\mathrm{r}}^{(n)} - \alpha^{(n)}A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\beta^{(n)} = \left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n+1)}, \boldsymbol{r}_{\mathrm{r}}^{(n+1)}\right) \Big/ \left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n)}, \boldsymbol{r}_{\mathrm{r}}^{(n)}\right)$$
endfor

Fig. 2. RR-KS algorithm equivalent to the one in Fig. 1. [This is derived from the first strategy and is identical to the one derived from the second strategy, that is, the CG algorithm with a folded preconditioner for (6)].

It should be noted that the second strategy does not necessarily ensure (23) nor (32) for the original preconditioned KS solver. However, it is quite unlikely that a sophisticated KS algorithm, such as the CG method, generates poor approximate solutions compared with the algorithm derived from the first strategy, because both algorithms search for approximate solutions in the same affine space. On the other hand, the second strategy may give a more sophisticated and efficient algorithm than the one derived from the first strategy.

As discussed below, the first strategy derives exactly the same algorithm as the second strategy for a certain class of KS methods, e.g. the CG method. This is not, however, the case for the GMRES method [13], for example.

B. Case in Which the Two Strategies Produce the Same RR-KS Algorithm

This subsection addresses the case in which the two strategies actually produce the same algorithm.

The CG method is one of the most sophisticated KS methods, and can be applied to problems involving positive definite matrices. Fig. 1 shows the preconditioned CG algorithm for (1).

According to the first strategy, we consider deriving an equivalent RR-KS algorithm to satisfy (23) from the CG algorithm for (1). We translate the original algorithm into a form that gives the recurrence formula with respect to $\boldsymbol{x}_{\mathrm{r}}'^{(n)} = \boldsymbol{x}_1^{(n)} + B\boldsymbol{x}_2^{(n)}$ (see the Appendix for details). Then, we apply the following substitutions:

$$\boldsymbol{x}_{\mathrm{r}}^{(n)} \leftarrow \boldsymbol{x}_{\mathrm{r}}'^{(n)} \tag{33}$$
$$\boldsymbol{r}_{\mathrm{r}}^{(n)} \leftarrow \boldsymbol{r}_{\mathrm{r}}'^{(n)} \tag{34}$$
$$\boldsymbol{p}_{\mathrm{r}}^{(n)} \leftarrow \boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)} \tag{35}$$

where

$$\boldsymbol{p}^{(n)} = \begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix}, \quad \boldsymbol{p}_1^{(n)} \in \boldsymbol{C}^L, \quad \boldsymbol{p}_2^{(n)} \in \boldsymbol{C}^{N-L}. \tag{36}$$

This gives the equivalent RR-KS algorithm shown in Fig. 2. This is identical to the standard CG algorithm for (6) using a folded preconditioner, which is directly derived from the second strategy.

For the CG method, we can again confirm the equality of the two strategies from the viewpoint of the orthogonal conditions. The preconditioned CG method for (1) finds the nth approximate solution [from the affine space represented by (13) and (14)] so as to satisfy the following orthogonal condition:

$$\left(M_1\boldsymbol{r}^{(n)}, M_1(AM)^{i-1}\boldsymbol{r}^{(0)}\right) = 0, \quad i = 1, 2, \ldots, n. \tag{37}$$

Using $M_1 = M_2^H$, which is due to the Hermitian property for the CG method, we obtain

$$\left(\boldsymbol{r}^{(n)}, M(AM)^{i-1}\boldsymbol{r}^{(0)}\right) = 0, \quad i = 1, 2, \ldots, n \tag{38}$$

that is

$$\left(\begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}^{(n)} \end{pmatrix}, \begin{pmatrix} (M_{11} + M_{12}C)(A_{\mathrm{r}}M_{\mathrm{f}})^{i-1}\boldsymbol{r}_{\mathrm{r}}^{(0)} \\ (M_{21} + M_{22}C)(A_{\mathrm{r}}M_{\mathrm{f}})^{i-1}\boldsymbol{r}_{\mathrm{r}}^{(0)} \end{pmatrix}\right) = 0, \quad i = 1, 2, \ldots, n. \tag{39}$$

Using $C = B^H$, which is again due to the Hermitian property of A, we obtain

$$\left(\boldsymbol{r}_{\mathrm{r}}^{(n)}, M_{\mathrm{f}}(A_{\mathrm{r}}M_{\mathrm{f}})^{i-1}\boldsymbol{r}_{\mathrm{r}}^{(0)}\right) = 0, \quad i = 1, 2, \ldots, n. \tag{40}$$

This is identical to the orthogonal condition that should be satisfied by the preconditioned CG method for (6).

Similarly, we can also confirm that the two strategies produce the same algorithms for the BiCG [14], COCG [15], CGS [16], and COCR [17] methods *inter alia*. The CR method [18] also belongs to this group, when it is applied to Hermitian matrices.

C. Case in Which the Two Strategies Produce Different RR-KS Algorithms

By contrast to the previous case, the two strategies produce different RR-KS algorithms for the CR (for non-Hermitian matrices), GMRES, GCR [18], BiCGSTAB [19], QMR [20], and TFQMR [21] methods *inter alia*.

In this paper, we present only the RR-KS algorithms based on the GCR method, because these are relatively simple. Fig. 3 shows the GCR algorithm for (1) with right-preconditioning, which means $M_1 = I$. Similar to the CG algorithm, substitutions (33)–(35) give the equivalent RR-KS algorithm shown in Fig. 4. This is different to the standard GCR algorithm with a folded preconditioner shown in Fig. 5.

What causes the difference between the algorithms derived from the two strategies? For example, the GCR method for (1) with right-preconditioning finds an approximate solution from the affine space (13) and (14) so as to minimize the residual norm

$$\left\|\boldsymbol{r}^{(n)}\right\| = \left\|\boldsymbol{r}_{\mathrm{r}}'^{(n)}\right\|_{I+C^HC}. \tag{41}$$

[Note (21).] This explains why the matrix $I+C^HC$ also appears in the algorithm presented in Fig. 4. Contrarily, the algorithm

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$$
$$\boldsymbol{p}^{(0)} = M\boldsymbol{r}^{(0)}$$
for $n = 1, 2, \ldots$
$$\alpha^{(n)} = \left(A\boldsymbol{p}^{(n)}, \boldsymbol{r}^{(n)}\right) \Big/ \left(A\boldsymbol{p}^{(n)}, A\boldsymbol{p}^{(n)}\right)$$
$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha^{(n)}\boldsymbol{p}^{(n)}$$
$$\boldsymbol{r}^{(n+1)} = \boldsymbol{r}^{(n)} - \alpha^{(n)}A\boldsymbol{p}^{(n)}$$
$$\beta^{(n,i)} = -\frac{\left(A\boldsymbol{p}^{(i)}, AM\boldsymbol{r}^{(n+1)}\right)}{\left(A\boldsymbol{p}^{(n)}, A\boldsymbol{p}^{(n)}\right)}, \quad i \le n$$
$$\boldsymbol{p}^{(n+1)} = M\boldsymbol{r}^{(n+1)} + \sum_{i=0}^{n}\beta^{(n,i)}\boldsymbol{p}^{(i)}$$
endfor

Fig. 3. Preconditioned GCR algorithm for (1).

$$\boldsymbol{r}_{\mathrm{r}}^{(0)} = \boldsymbol{b}_{\mathrm{r}} - A\boldsymbol{x}_{\mathrm{r}}^{(0)}$$
$$\boldsymbol{p}_{\mathrm{r}}^{(0)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(0)}$$
for $n = 1, 2, \ldots$
$$\alpha^{(n)} = \left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, \boldsymbol{r}_{\mathrm{r}}^{(n)}\right)_{I+C^HC} \Big/ \left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}\right)_{I+C^HC}$$
$$\boldsymbol{x}_{\mathrm{r}}^{(n+1)} = \boldsymbol{x}_{\mathrm{r}}^{(n)} + \alpha^{(n)}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\boldsymbol{r}_{\mathrm{r}}^{(n+1)} = \boldsymbol{r}_{\mathrm{r}}^{(n)} - \alpha^{(n)}A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\beta^{(n,i)} = -\frac{\left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(i)}, A_{\mathrm{r}}M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n+1)}\right)_{I+C^HC}}{\left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}\right)_{I+C^HC}}, \quad i \le n$$
$$\boldsymbol{p}_{\mathrm{r}}^{(n+1)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n+1)} + \sum_{i=0}^{n}\beta^{(n,i)}\boldsymbol{p}_{\mathrm{r}}^{(i)}$$
endfor

Fig. 4. RR-KS algorithm equivalent to the one in Fig. 3. (This is derived from the first strategy and is *not* identical to the one derived from the second strategy.)

$$\boldsymbol{r}_{\mathrm{r}}^{(0)} = \boldsymbol{b}_{\mathrm{r}} - A\boldsymbol{x}_{\mathrm{r}}^{(0)}$$
$$\boldsymbol{p}_{\mathrm{r}}^{(0)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(0)}$$
for $n = 1, 2, \ldots$
$$\alpha^{(n)} = \left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, \boldsymbol{r}_{\mathrm{r}}^{(n)}\right) \Big/ \left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}\right)$$
$$\boldsymbol{x}_{\mathrm{r}}^{(n+1)} = \boldsymbol{x}_{\mathrm{r}}^{(n)} + \alpha^{(n)}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\boldsymbol{r}_{\mathrm{r}}^{(n+1)} = \boldsymbol{r}_{\mathrm{r}}^{(n)} - \alpha^{(n)}A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}$$
$$\beta^{(n,i)} = -\frac{\left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(i)}, A_{\mathrm{r}}M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n+1)}\right)}{\left(A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}, A_{\mathrm{r}}\boldsymbol{p}_{\mathrm{r}}^{(n)}\right)}, \quad i \le n$$
$$\boldsymbol{p}_{\mathrm{r}}^{(n+1)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}^{(n+1)} + \sum_{i=0}^{n}\beta^{(n,i)}\boldsymbol{p}_{\mathrm{r}}^{(i)}$$
endfor

Fig. 5. RR-KS algorithm derived from the second strategy, that is, the GCR algorithm with a folded preconditioner for (6).

shown in Fig. 5 minimizes $||\boldsymbol{r}_{\mathrm{r}}^{(n)}||$. The other KS methods cited above are also (partly) based on the minimization of the residual norm to decide the parameters in their own algorithms. This causes the difference between the derived algorithms.

Minimization of $||\boldsymbol{r}_{\mathrm{r}}^{(n)}||$ is considered reasonable and proper from the viewpoint of solving (6), especially when the terminating condition of an iteration is stated with respect to $||\boldsymbol{r}_{\mathrm{r}}^{(n)}||$. It is unlikely that the sophisticated KS algorithms will choose poor approximate solutions from the same affine space, which is improved by the folded preconditioning. On the other hand, the first strategy tends to lead to more complicated and expensive algorithms than the second one. For example, the algorithm in Fig. 4 needs additional matrix-vector multiplications with respect to $I + C^HC$, compared with the algorithm in Fig. 5. We recommend the second strategy for practical use.

IV. Computational Procedures for the Folded Preconditioners and Efficiency of RR-KS Solvers

A. Computational Procedures for the Folded Preconditioners

The remaining task is to design computational procedures for the folded preconditioners. To implement the algorithms described in the previous section, the following matrix-vector multiplication needs to be computed:

$$\boldsymbol{q}_{\mathrm{r}} = M_f\boldsymbol{r}_{\mathrm{r}} = (M_{11} + M_{12}C + BM_{21} + BM_{22}C)\boldsymbol{r}_{\mathrm{r}} \quad (42)$$

where the superscript (i) is omitted. Although a certain class of preconditioners, e.g., the Incomplete LU (ILU) factorization, does not explicitly give matrix M, we can at least obtain $\boldsymbol{q}_{\mathrm{r}}$ from the following procedure.

Procedure 1: General Procedure for Folded Preconditioners: Because M is used as the inverse of a preconditioner for (1), we must be able to compute the following vector, whether M is given explicitly or not

$$\begin{pmatrix}\boldsymbol{q}_1\\ \boldsymbol{q}_2\end{pmatrix} = \begin{pmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{pmatrix}\begin{pmatrix}\boldsymbol{r}_{\mathrm{r}}\\ C\boldsymbol{r}_{\mathrm{r}}\end{pmatrix}. \quad (43)$$

Then, the vector $\boldsymbol{q}_{\mathrm{r}}$ is obtained from

$$\boldsymbol{q}_{\mathrm{r}} = \boldsymbol{q}_1 + B\boldsymbol{q}_2. \quad (44)$$

□

Consequently, for any original preconditioner, we can compute $\boldsymbol{q}_{\mathrm{r}}$ with at most i) the same costs as in the original preconditioning plus ii) additional costs from the matrix-vector multiplications with respect to B and C.

Moreover, there is still room for improvement with respect to the individual implementations. The following subsections address the cases in which the original preconditioners are the ILU factorization and Gauss-Seidel (GS) method.

B. Folded ILU (IC) Preconditioner

The ILU preconditioner for (1) is given in the form

$$M^{-1} = \begin{pmatrix}L_{11} & \\ L_{21} & L_{22}\end{pmatrix}\begin{pmatrix}U_{11} & U_{12}\\ & U_{22}\end{pmatrix}. \quad (45)$$

As mentioned in the previous subsection, the matrix-vector multiplication with respect to the folded ILU preconditioner can be implemented by Procedure 1. Moreover, we can improve it as follows.

Since

$$\begin{pmatrix}L_{11} & \\ L_{21} & L_{22}\end{pmatrix}^{-1} = \begin{pmatrix}L_{11}^{-1} & \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1}\end{pmatrix}$$
$$\begin{pmatrix}U_{11} & U_{12}\\ & U_{22}\end{pmatrix}^{-1} = \begin{pmatrix}U_{11}^{-1} & -U_{11}^{-1}U_{12}U_{22}^{-1}\\ & U_{22}^{-1}\end{pmatrix} \quad (46)$$

after some calculation, we can represent the inverse of the folded IC preconditioner as

$$M_{\rm f} = U_{11}^{-1}L_{11}^{-1} + \left(B - U_{11}^{-1}U_{12}\right)U_{22}^{-1}L_{22}^{-1}\left(C - L_{21}L_{11}^{-1}\right). \tag{47}$$

This can be rewritten as

$$M_{\rm f} = U_{11}^{-1}\left(I + U'_{12}U_{22}^{-1}L_{22}^{-1}L'_{21}\right)L_{11}^{-1} \tag{48}$$

where

$$L'_{21} = L_{21} - CL_{11}, \quad U'_{12} = U_{12} - U_{11}B. \tag{49}$$

Using (48) and (49), we can implement the folded ILU preconditioner by the following procedure.

Procedure 2: Modified Procedure for the Folded ILU Preconditioner: To obtain $\boldsymbol{q}_{\rm r} = M_{\rm f}\boldsymbol{r}_{\rm r}$, compute

$$\begin{pmatrix} U_{11} & U'_{12} \\ & U_{22} \end{pmatrix}^{-1}\begin{pmatrix} L_{11} & \\ L'_{21} & L_{22} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{r}_{\rm r} \\ 0 \end{pmatrix} = \begin{pmatrix} U_{11}^{-1}\left(I + U'_{12}U_{22}^{-1}L_{22}^{-1}L'_{21}\right)L_{11}^{-1}\boldsymbol{r}_{\rm r} \\ -U_{22}^{-1}L_{22}^{-1}L'_{21}L_{11}^{-1}\boldsymbol{r}_{\rm r} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_{\rm r} \\ * \end{pmatrix} \tag{50}$$

using the (standard) forward and backward substitution. □

Assuming that we apply the ILU factorization prohibiting fill-ins, L_{11} and L_{21} have the same sparse patterns as A and CA, respectively. Noting (49), L'_{21} is expected to have nearly the same sparse pattern as L_{21} and similarly U_{12} as U'_{12}. In fact, $L'_{21}(U'_{12})$ has exactly the same sparse pattern as $L_{21}(U_{12})$ in the full-wave analysis addressed in Section VI. Overwriting $L_{21}(U_{12})$ by $L'_{21}(U'_{12})$ after the ILU factorization, we can execute the folded ILU preconditioning with nearly the same costs as the original ILU preconditioning.

When the original coefficient matrix has the symmetric property, we obtain the folded incomplete Cholesky (IC) preconditioner.

C. Folded GS Preconditioner

The (forward) GS preconditioner for (1) is given by

$$M^{-1} = \begin{pmatrix} L_1 & \\ CA_{\rm r} & L_2 \end{pmatrix} \tag{51}$$

where L_1 and L_2 are the lower triangular parts of $A_{\rm r}$ and $CA_{\rm r}B$, respectively.

The folded GS preconditioner is represented by

$$M_{\rm f} = L_1^{-1} + BL_2^{-1}C\left(I - A_{\rm r}L_1^{-1}\right). \tag{52}$$

Instead of using Procedure 1, we have an alternative way of obtaining $\boldsymbol{q}_{\rm r} = M_{\rm f}\boldsymbol{r}_{\rm r}$.

Procedure 3: Modified Procedure for the Folded GS Preconditioner: First compute $\boldsymbol{u} = L_1^{-1}\boldsymbol{r}_{\rm r}$ by a GS sweep and obtain the vector $\boldsymbol{r}_{\rm n} = C(L_1 - A_{\rm r})\boldsymbol{u}$. Then compute $\boldsymbol{v} = L_2^{-1}\boldsymbol{r}_{\rm n}$ by a GS sweep with respect to the matrix $CA_{\rm r}B$. Finally, obtain $\boldsymbol{q}_{\rm r} = \boldsymbol{u} + B\boldsymbol{v}$. □

Although Procedure 3 requires matrix-vector multiplications with respect to B, C and $(L_1 - A_{\rm r})$, it omits the multiplication with respect to $CA_{\rm r}$, which is required in Procedure 1.

D. Efficiency of RR-KS Solvers With Folded Preconditioners

Here we discuss the efficiency of the RR-KS solver with the folded preconditioner. In general, it is required that the number of reduced DOF is sufficiently large and that matrices B and C are sparse and given explicitly, to attain better performance than the original preconditioned KS solver for (1). If this is the case, the performance of the RR-KS solver is promising, because it is expected that

i) the convergence of the RR-KS solver is equal to that of the original solver for (1), at least when we adopt the first strategy in Section III;
ii) the RR-KS solver substantially reduces the computation costs with respect to the matrix-vector multiplications and other vector computations in the iteration except for the preconditioning;
iii) the folded preconditioner using Procedure 1 involves additional effort only to compute the multiplications with respect to B and C, compared with the original preconditioner.

Moreover, we can improve the individual implementation of the folded preconditioners, e.g., by using Procedures 2 and 3.

More specifically, consider the CG solver for (6) with the folded IC preconditioner (the folded ICCG solver) using Procedure 2. The folded ICCG solver is robustly superior to the original ICCG solver for (1), because the folded ICCG solver is equivalent to the ICCG solver for (1) and the computation cost for the folded IC preconditioner using Procedure 2 is nearly the same as the original IC preconditioner for (1).

V. Application to Finite Element Analysis Using the A-φ Formulation

In this section the application to finite element analysis is discussed. In full-wave analysis, the A-φ formulation leads to the linear system of equations [4], in which

$$[A_{\rm r}]_{lm} = \int \nu(\nabla \times \boldsymbol{N}_l)\cdot(\nabla \times \boldsymbol{N}_m)dV - \omega^2\int\left(\varepsilon + \frac{\sigma}{j\omega}\right)\boldsymbol{N}_l\cdot\boldsymbol{N}_m dV \tag{53}$$

and

$$B = G, \quad C = G^T. \tag{54}$$

Here, j, $\boldsymbol{N}$, ν, ε, σ, ω, and G denote an imaginary unit, the edge-element basis functions, magnetic reluctivity, electric permittivity, electric conductivity, exciting angular frequency, and the discrete gradient operator [4], [10], respectively.

Because B and C are sparse and explicitly given, we can construct efficient RR-KS solvers with the folded preconditioners. For example, we can apply the COCG solver with the folded IC preconditioner (the folded ICCOCG solver), which has the same convergence property as the ICCOCG solver for the original equations arising from the A-φ formulation. The performance of the folded ICCOCG solver using Procedure 2 is promising, because it surely involves less computation cost per iteration than the standard ICCOCG solver for the original equations.

Similarly, the folded ICCG and ICCOCG methods are efficient solvers for eddy-current analysis using the A-φ formulation.

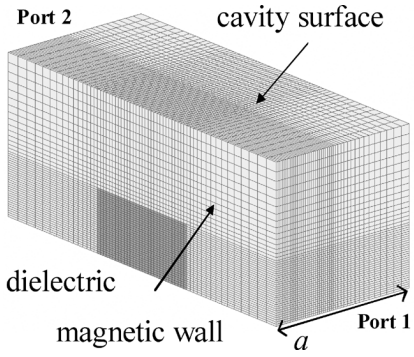

Fig. 6. Dielectric loaded waveguide.

TABLE I
NUMBERS OF DOF AND NONZERO ENTRIES OF THE COEFFICIENT MATRIX

Formulations	A	A-φ
Number of DOF	363,107	483,887
Number of Nonzero Entries (Upper Triangular Part)	6,006,199	13,930,984

VI. NUMERICAL TESTS

In this section, the performance of the folded ICCOCG solver is examined in a sample full-wave analysis. Fig. 6 shows the test model [3], a dielectric loaded waveguide discretized by rectangular elements (1/2 model). The relative permittivity of the dielectric material is set to 6.0-j 1.0. The angular frequency is decided such that $k_0 a = 2.6$, where k_0 and a denote the free-space wave number and one-half width of the waveguide, respectively.

Table I shows the numbers of DOF and nonzero entries of the coefficient matrix. The number of redundant DOF in the A-φ formulation is 120 780.

The acceleration parameter in the IC factorization is set to 1.2. Computations are executed on a Workstation (Xeon X5472, 8 GB RAM). The convergence criterion for the RR-KS solvers is

$$\left\|\boldsymbol{r}_{\mathrm{r}}^{(n)}\right\| \leq \varepsilon \left\|\boldsymbol{r}_{\mathrm{r}}^{(0)}\right\| \tag{55}$$

where ε is 1e-8. For fair comparison

$$\left\|\boldsymbol{r}_{\mathrm{r}}^{\prime(n)}\right\| \leq \varepsilon \left\|\boldsymbol{r}_{\mathrm{r}}^{\prime(0)}\right\| \tag{56}$$

is used as the criterion for the KS solver for the redundant systems arising from the A-φ formulation.

We apply the standard and folded ICCOCG solvers to a redundant system and redundancy-reduced system, respectively. Fig. 7 compares the convergence of $\|\boldsymbol{r}_{\mathrm{r}}^{(n)}\|$ and $\|\boldsymbol{r}_{\mathrm{r}}^{\prime(n)}\|$ with respect to both solvers. As predicted, the profiles of these norms nearly coincide, although there is a slight difference caused by round-off errors. This is due to the mathematical equivalence of both solvers.

Table II depicts the performance of the iterative solvers, including the COCG solver for the redundancy-reduced system using the standard IC preconditioner ("ICCOCG (A)" in the table). Whereas the reduction in DOF causes significant deterioration in the convergence property when using the standard IC preconditioner, the folded IC preconditioner counteracts the effect of the redundancy-reduction on the convergence property. This results in the good performance of the new solvers with respect to elapsed time. The new solvers can also reduce the memory consumption, compared with the ICCOCG solver for the A-φ formulation. Moreover, it is confirmed that the performance of the folded ICCOCG solver is improved considerably by using Procedure 2.

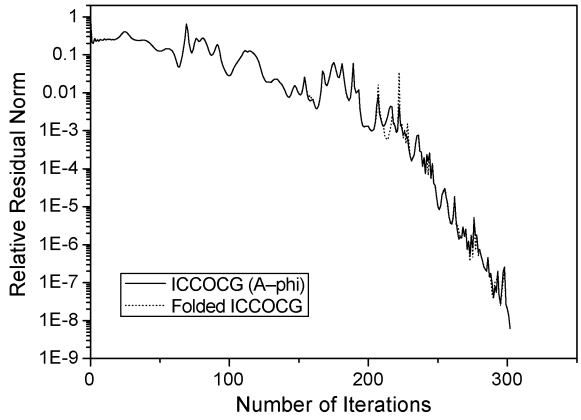

Fig. 7. Comparison of convergence behavior of the ICCOCG method for the A-φ formulation and the folded ICCOCG method. Note that this figure plots the relative residual norms with respect to the redundancy-reduced system. See (56).

TABLE II
PERFORMANCE OF THE ITERATIVE SOLVERS

Solver	ICCOCG (A-φ)	ICCOCG (A)	Folded ICCOCG (Procedure 1)	Folded ICCOCG (Procedure 2)
Number of Iterations	302	1772	302	302
Elapsed Time (s)	98.8	265.5	89.4	81.8
Maximum Memory Consumption (MB)	663 (550)	356 (260)	542 (444)	524 (427)

Figures in parentheses express the memory consumptions (calculated values) required to execute the preconditioned COCG solvers.

VII. CONCLUSION

We have proposed the folded preconditioners: a new class of preconditioners for the RR-KS methods, which is useful for solving redundant linear systems of equations efficiently. The theorem we present reveals that, for an arbitrary preconditioned KS method for the original redundant systems, the equivalent RR-KS method can be constructed using the corresponding folded preconditioner. We can reduce the redundant DOF and the computation costs with respect to the matrix-vector multiplications and other vector computations in the KS algorithm, without any deterioration in the convergence property.

The folded preconditioner can be implemented efficiently irrespective of the original preconditioner, if matrices B and C are sparse and given explicitly. Furthermore, for specific preconditioners, such as the IC factorization, we are able to improve the implementation. For instance, the folded IC preconditioner needs approximately the same computational effort as the original IC preconditioner.

For practical applications, the folded ICCG and ICCOCG methods are promising solvers for full-wave or eddy-current analysis. These methods provide a solution to the problem of

$$\begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(0)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(0)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{\mathrm{r}} \\ C\boldsymbol{b}_{\mathrm{r}} \end{pmatrix} - \begin{pmatrix} A_{\mathrm{r}} & A_{\mathrm{r}}B \\ CA_{\mathrm{r}} & CA_{\mathrm{r}}B \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1^{(0)} \\ \boldsymbol{x}_2^{(0)} \end{pmatrix}, \quad \beta_{-1} = 0$$

for $n = 0, 1, \ldots$

$$\begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} + \beta^{(n-1)} \begin{pmatrix} \boldsymbol{p}_1^{(n-1)} \\ \boldsymbol{p}_2^{(n-1)} \end{pmatrix}$$

$$\alpha^{(n)} = \frac{\left(\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} \right)}{\left(\begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix}, \begin{pmatrix} A_{\mathrm{r}} & A_{\mathrm{r}}B \\ CA_{\mathrm{r}} & CA_{\mathrm{r}}B \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix} \right)}$$

$$\begin{pmatrix} \boldsymbol{x}_1^{(n+1)} \\ \boldsymbol{x}_2^{(n+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{(n)} \\ \boldsymbol{x}_2^{(n)} \end{pmatrix} + \alpha^{(n)} \begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} - \alpha^{(n)} \begin{pmatrix} A_{\mathrm{r}} & A_{\mathrm{r}}B \\ CA_{\mathrm{r}} & CA_{\mathrm{r}}B \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{p}_2^{(n)} \end{pmatrix}$$

$$\beta^{(n)} = \frac{\left(\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n+1)} \end{pmatrix} \right)}{\left(\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{r}_{\mathrm{r}}'^{(n)} \\ C\boldsymbol{r}_{\mathrm{r}}'^{(n)} \end{pmatrix} \right)}$$

endfor

Fig. 8. Rewritten CG algorithm.

$$\boldsymbol{r}_{\mathrm{r}}'^{(0)} = \boldsymbol{b}_{\mathrm{r}} - A_{\mathrm{r}}\left(\boldsymbol{x}_1^{(0)} + B\boldsymbol{x}_2^{(0)}\right), \quad \beta_{-1} = 0$$

for $n = 0, 1, \ldots$

$$\boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)} = M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(n)} + \beta^{(n-1)}\left(\boldsymbol{p}_1^{(n-1)} + B\boldsymbol{p}_2^{(n-1)}\right)$$

$$\alpha^{(n)} = \frac{\left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(n)}, \boldsymbol{r}_{\mathrm{r}}'^{(n)}\right)}{\left(\left(\boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)}\right), A_{\mathrm{r}}\left(\boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)}\right)\right)}$$

$$\boldsymbol{x}_1^{(n+1)} + B\boldsymbol{x}_2^{(n+1)} = \boldsymbol{x}_1^{(n)} + B\boldsymbol{x}_2^{(n)} + \alpha^{(n)}\left(\boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)}\right)$$

$$\boldsymbol{r}_{\mathrm{r}}'^{(n+1)} = \boldsymbol{r}_{\mathrm{r}}'^{(n)} - \alpha^{(n)}A_{\mathrm{r}}\left(\boldsymbol{p}_1^{(n)} + B\boldsymbol{p}_2^{(n)}\right)$$

$$\beta^{(n)} = \frac{\left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(n+1)}, \boldsymbol{r}_{\mathrm{r}}'^{(n+1)}\right)}{\left(M_{\mathrm{f}}\boldsymbol{r}_{\mathrm{r}}'^{(n)}, \boldsymbol{r}_{\mathrm{r}}'^{(n)}\right)}$$

endfor

Fig. 9. Equivalent algorithm to the one in Fig. 8, which gives the recurrence formula with respect to $\boldsymbol{x}_{\mathrm{r}}'^{(n)} = \boldsymbol{x}_1^{(n)} + B\boldsymbol{x}_2^{(n)}$.

poor convergence in conventional solvers caused by the redundancy reduction in the A-φ formulation.

The folded preconditioners have a strong relationship to the explicit/implicit error correction methods [11] and the singularity decomposition technique [22]. In fact, Procedure 3 in Section IV can be regarded as a kind of explicit error correction method based on the GS method. The relationship between these methods will be discussed in future works.

APPENDIX

To derive the algorithm in Fig. 2 using the first strategy, we rewrite the CG algorithm (Fig. 1) as shown in Fig. 8, using (20), (21), and (36).

Noting $C = B^H$, which is due to the Hermitian property for the CG method, we obtain the algorithm shown in Fig. 9. This gives the recurrence formula with respect to $\boldsymbol{x}_{\mathrm{r}}'^{(n)} = \boldsymbol{x}_1^{(n)} + B\boldsymbol{x}_2^{(n)}$ in the form of an RR-KS algorithm.

Finally, applying substitutions (33)–(35), we obtain the algorithm presented in Fig. 2.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*. Cambridge, U.K.: Cambridge Univ. Press, 1994.

[2] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Philadelphia, PA: SIAM, 1997.

[3] R. Edlinger and O. Biro, "A joint vector and scalar potential formulation for driven high-frequency problems using hybrid edge and nodal finite elements," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 15–23, Jan. 1995.

[4] Y. Zhu and A. C. Cangellaris, *Multigrid Finite Element Methods for Electromagnetic Field Modeling*. New York: IEEE Press, 2006, pp. 61–96.

[5] J. Jin, *The Finite Element Method in Electromagnetics*, 2nd ed. New York: Wiley, 2002.

[6] J. B. Manges and Z. Cendes, "A generalized tree-cotree gauge for magnetic field computation," *IEEE Trans. Magn.*, vol. 31, no. 3, pp. 1342–1347, May 1995.

[7] K. Fujiwara, T. Nakata, and H. Ohashi, "Improvement of convergence characteristics of ICCG method for the A-j method using edge elements," *IEEE Trans. Magn.*, vol. 32, pp. 804–807, 1996.

[8] H. Igarashi, "On the property of the curl-curl matrix in finite element analysis with edge elements," *IEEE Trans. Magn.*, vol. 37, no. 5, pp. 3129–3132, Sep. 2001.

[9] H. Igarashi and T. Honma, "On convergence of ICCG applied to finite element equation for quasi-static fields," *IEEE Trans. Magn.*, vol. 38, pp. 565–568, 2002.

[10] H. Igarashi and N. Yamamoto, "Effect of reconditioning in edge-based finite-element method," *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 942–945, Jun. 2008.

[11] T. Iwashita, T. Mifune, and M. Shimasaki, "Similarities between implicit correction multigrid method and A-phi formulation in electromagnetic field analysis," *IEEE Trans. Magn.*, vol. 44, pp. 946–949, 2008.

[12] M. R. Hestenes and E. Steifel, "Method of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Stand.*, vol. 49, pp. 409–436, Dec. 1952.

[13] Y. Saad, "GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 856–869, 1986.

[14] C. Lanczos, "Solution of systems of linear equations by minimized iterations," *J. Res. Nat. Bur. Stand.*, vol. 49, pp. 33–53, 1952.

[15] H. A. van der Vorst and J. B. M. Melissen, "A Petrov-Galerkin type method for solving $Ax = b$, where A is symmetric complex," *IEEE Trans. Magn.*, vol. 26, no. 2, pp. 706–708, Mar. 1990.

[16] P. Sonneveld, "CGS: A fast Lanczos-type solver for nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 10, pp. 36–52, 1989.

[17] T. Sogabe and S. Zhang, "A COCR method for solving complex symmetric linear systems," *J. Comput. Appl. Math.*, vol. 199, no. 2, pp. 297–303, Feb. 2007.

[18] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, "Variational iterative methods for nonsymmetric systems of linear equations," *SIAM J. Numer. Anal.*, vol. 20, no. 2, pp. 345–357, 1983.

[19] H. A. van der Vorst, "Bi-CGSTAB: A more smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 13, pp. 631–644, 1992.

[20] R. W. Freund and N. M. Nachtigal, "QMR: A quasi-minimal residual method for non-Hermitian linear systems," *Numer. Math.*, vol. 60, pp. 315–339, 1991.

[21] R. W. Freund, "A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems," *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 470–482, 1993.

[22] A. Kameari, "Improvement of ICCG convergence for thin elements in magnetic field analyses using the finite-element method," *IEEE Trans. Magn.*, vol. 44, pp. 1178–1181, 2008.